#SHAREorg

SHARE

DB2 11 *NEW* Availability Functions and Features

John Iczkovits
IBM (iczkovit@us.ibm.com)

March 13, 2014
15276

Test link: www.SHARE.org

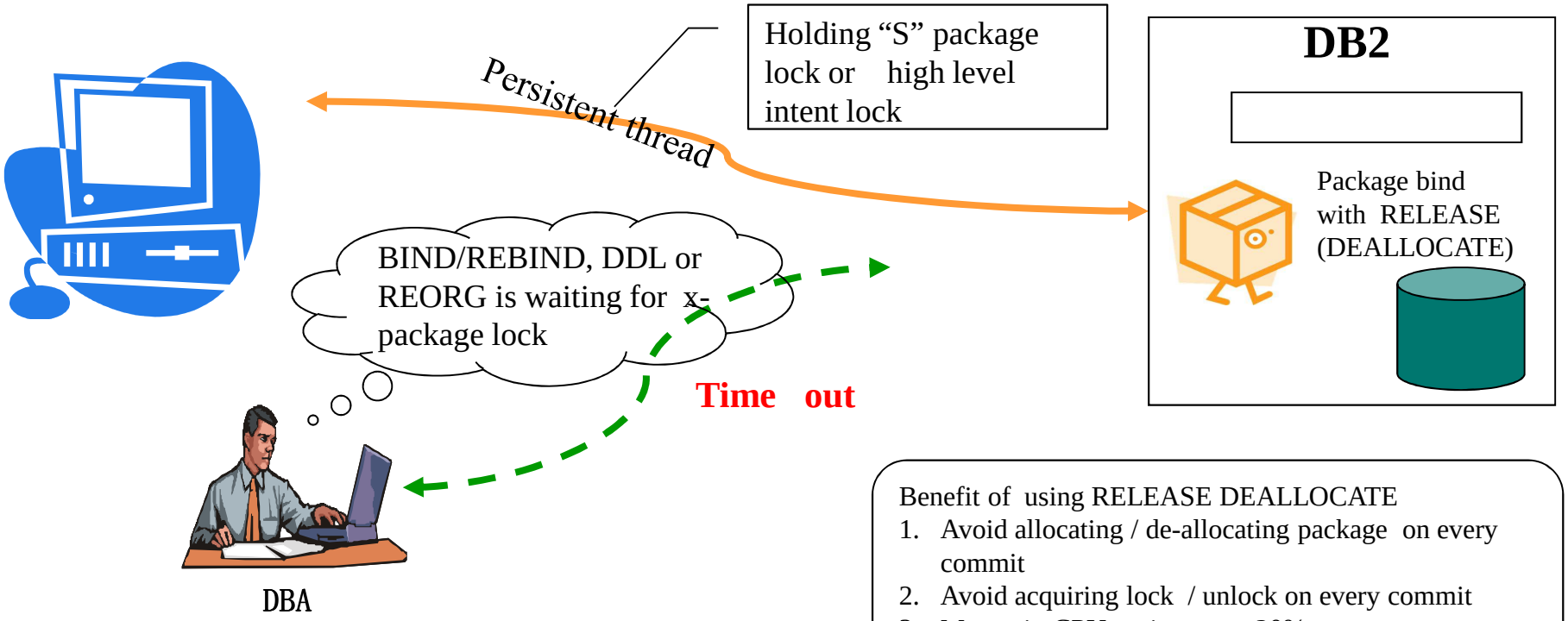
SHARE in Anaheim

Agenda

- Availability for BIND/REBIND/DDL to break-in persistent threads
- Availability enhancement in On-line schema changes
  - Drop Column
  - Online Alter Partition Limit Keys
  - Allow PIT recovery before materialized deferred alters
- Defer Define Object Enhancement
- Work file Enhancement
- Save old compression dictionary to support IFICID 306
- Avoid rebuild index after LPL/GRECP recovery

BIND/REBIND/DDL and Persistent Threads user story (prior to DB2 11)

- **A persistent thread that has package bind with RELEASE (DEALLOCATE) accessing DB2**



Benefit of using RELEASE DEALLOCATE
1. Avoid allocating / de-allocating package on every commit
2. Avoid acquiring lock / unlock on every commit
3. May gain CPU saving up to 20%

- **As a DBA, I need to deploy a new application which requires BIND REPLACE or REBIND PACKAGE**
- **As a DBA, I need to run DDL or online REORG to materialize pending ALTERS for tables/indexes accessed by the persistent threads**

Note : RELEASE(DEALLOCATE) Packages Behavior Prior To DB2 11

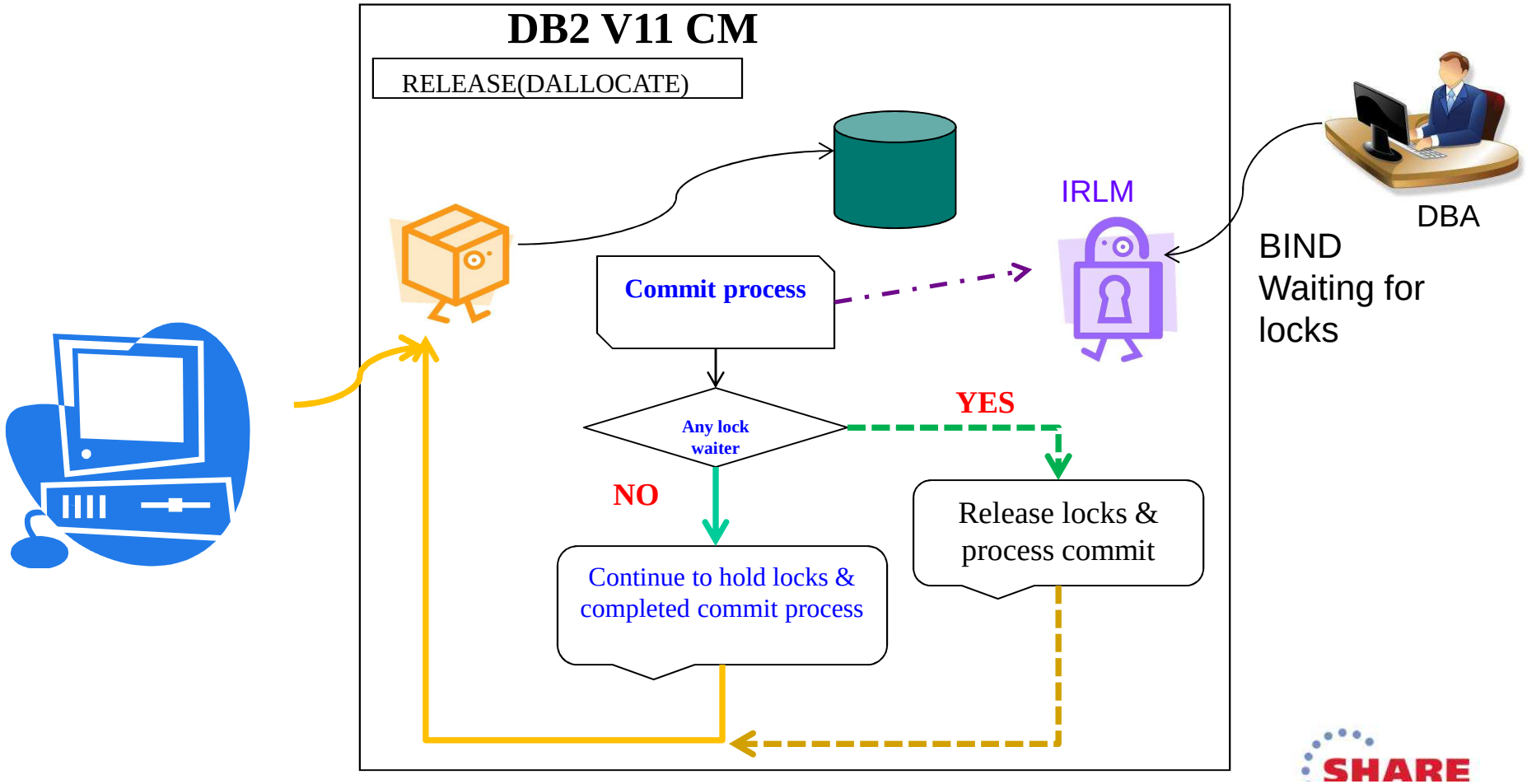
- Used for frequently accessed performance sensitive applications
  - CICS/IMS persistent threads
  - High performance data base application threads in DB2
- Used to avoid allocating and de-allocating package on every commit
  - S package locks will be kept until the thread is terminated
- Use to avoid acquiring and releasing high level locks
  - Such as table space/table/partition intent locks on every commit
  - Table space/table/partition intent locks are held until thread terminated
- As the result, CPU saving can be up to 20% compared to RELEASE(COMMIT)

NOTE: RELEASE(DEALLOCATE) Packages side effects prior to DB2 11

- Requires more thread storage
  - 90-95% thread storage has been moved above the bar in DB2 10
- BIND/REBIND can't break in due to S package locks
  - BIND/REBIND acquire X package locks to quiesce applications
- DDL or REORG utility can't break in
  - Due to incomparable with S package lock or
  - Due to incomparable table/partition intent lock
- Any operation has contention to the package or intent lock requires the persistent thread to be cancelled

DB2 V11 CM allows break-in Behavior

- **A persistent thread that has package bind with RELEASE (DEALLOCATE) accessing DB2**



- A new ZPARM parameter PKGREL_COMMIT with YES as the default

Allow Break-in to Persistent Thread behavior

- Starts V11 CM, DB2 allows BIND/REBIND/DDL/REORG to break into RELEASE(DEALLOCATE) packages accessed by persistent thread
- DB2 handles break-in during commit process of persistent thread that has RELEASE(DEALLOCATE) package
  - DB2 uses a IRLM fast path to query outstanding request
  - If there is a waiter, temporary switch the package to RELEASE(COMMIT)
    - Release S package lock
    - Release table space/table/partition locks if table is not accessed by other
    - De-allocate package
  - When package resumes after commit, it is being accessed as RELEASE(DEALLOCATE) again
- A new ZPARM parameter PKGREL_COMMIT with YES as the default

Break into Idle thread

- Idle thread does not entering commit process, so it can not detect waiter
- With the following apars, bind and DDL will be much more likely to break into a system with idle threads …
  - PM95929 – V11 early code support

    If this is not applied, break in for idle threads is not supported
  - PM96001 – V11 DB2 toleration code

    This must be applied to all members prior to applying PM96004
  - PM96004 – V11 DB2 enabling code.

    Once PM96001 applied to all members, this may be applied to any member.

    Without this applied to all members, break in may not be possible but will still be attempted.
- Available in NFM and **with V11** early code at the required maintenance level

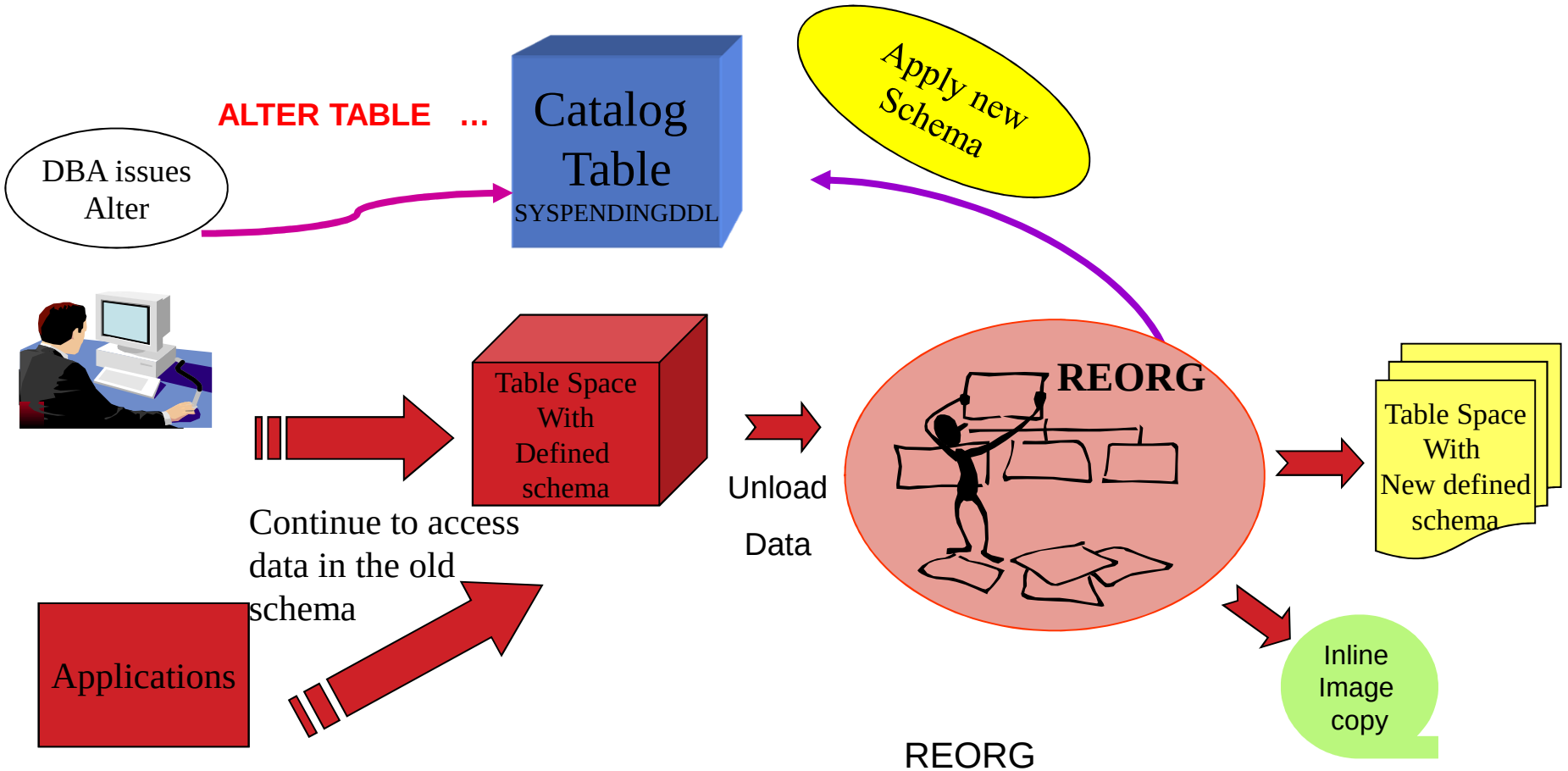
Things to think about

- Package with RELEASE(DEALLOCATE) cannot allow break in If…
  - Package bound with KEEPDYNAMIC=YES
  - Package has open and held cursor
  - Commit issued within a stored procedure
- Detecting waiter (release package lock and table space/table/index intent lock) is during the **commit process** of the persistent thread
  - Frequent commit will help with the break in
- May help by allowing longer waiting period for break-in to happen
  - Increase DDL timeout value (zparm **DDLTOX** / DATA DEF TIMEOUT)
  - IRLM timeout value (zparm **IRLMRWT** / RESOURCE TIMEOUT)
  - Caution – zparm DDLTOX and IRLMRWT are system parameter
- Still possible to get resource not available
  - Commit is not done frequently
  - Wait time expired

Online Schema Changes

- **Availability enhancement - Online schema changes**
  - **Drop Column**
  - **Online Alter Partition Limit Keys**
  - **Allow PIT recovery before materialized deferred alters**

Overview of Pending Alter



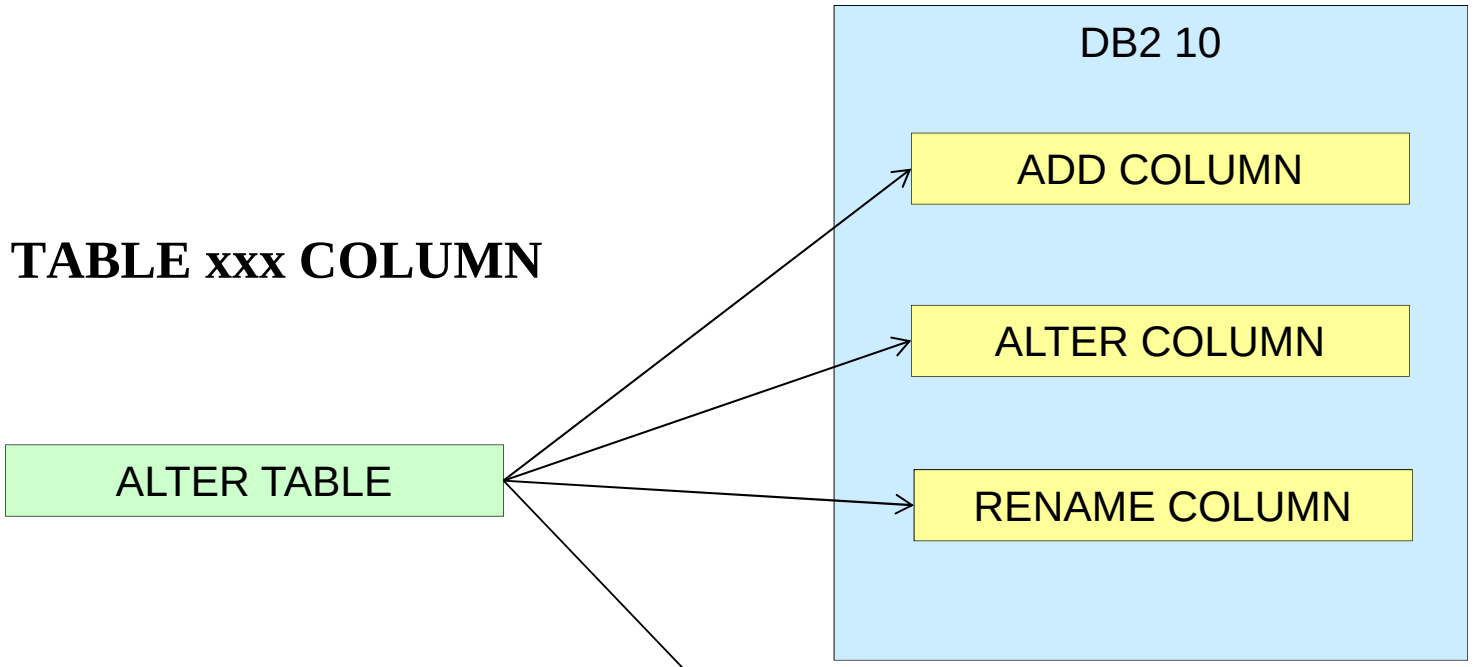
- At ALTER time :
- authorization checks
- SYSPENDINGDDL entries
- Table space set in AREOR

DB2 10 Online Schema Enhancements

Pending Change	PBG UTS	PBR UTS	XML TS	LOB TS
Alter Segment Size	X	X	X	
Alter Data Set Size	X	X	X	X
Alter BP T/S Page Size	X	X		X
Alter BP Index Page	X	X		
Alter Member Cluster	X	X		
Convert Classic 1- Table Part. T/S to UTS PBR	X	X		
Convert single table Classic Simple T/S	X			
Convert single table Classic Segmented T/S	X			

Changes on column level of existing tables

ALTER TABLE xxx COLUMN



sc1.table1 in DB1.TS1		
Col_1	Col_2	Col_3
Data_1	Data_2	Data_3
Data_1	Data_2	Data_3

Why DROP COLUMN?

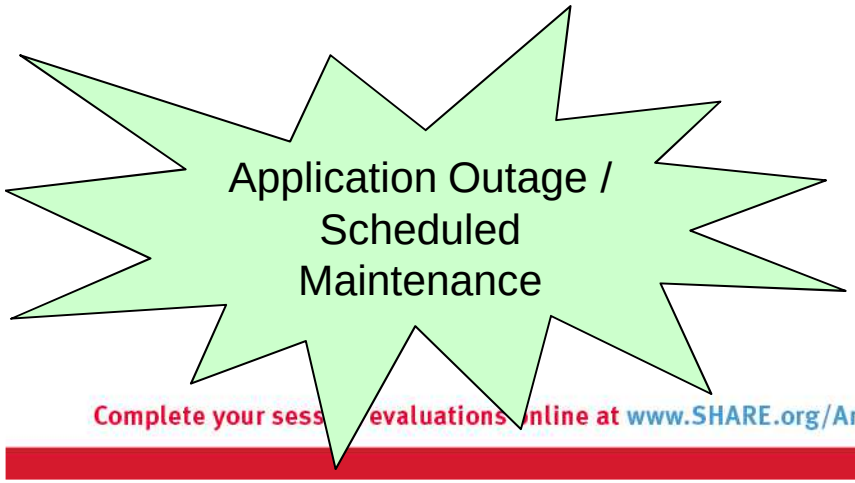
- Columns become redundant as applications change
- DBA, error on CREATE or ALTER TABLE
- Could the column be left?
  - An abandoned column costs
    - Real space in every row stored in the table
    - Real space in every Image Copy of the table space
    - Potential space taken up in the log records written for the table
    - Additional CPU and elapsed time in all aspects of accessing and maintaining the data
    - DBA time “remembering” that the column is redundant
    - Developer time analyzing usage of the column

sc1.table1 in DB1.TS1			
Col_1	**Col_2**	**Col_3**	**Col_4**
Data_1	Data_2	Data_3	Data_4
Data_1	Data_2	Data_3	Data_4

How DROP COLUMN works

- Before DB2 V11
  - You have to DROP and recreate the Table to remove Col_3

sc1.table1 in DB1.TS1			
Col_1	**Col_2**	**Col_3**	**Col_4**
Data_1	Data_2	Data_3	Data_4
Data_1	Data_2	Data_3	Data_4



sc1.table1 in DB1.TS1		
Col_1	**Col_2**	**Col_4**
Data_1	Data_2	Data_4
Data_1	Data_2	Data_4

Improvements in DB2 11 – DROP COLUMN

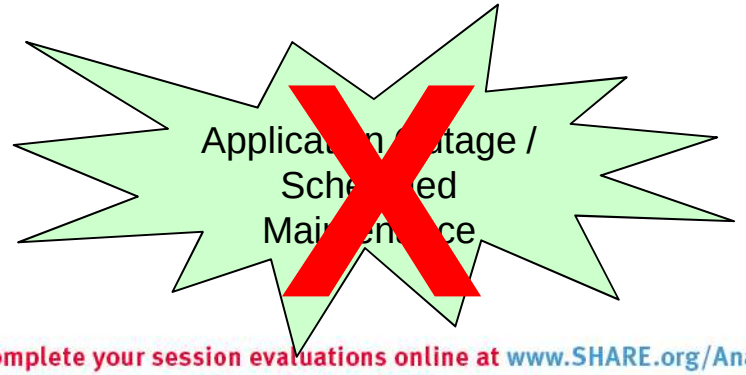
- DB2 11 NFM

```
ALTER TABLE sc1.table1 DROP COLUMN Col_3 RESTRICT
REORG TABLESPACE DB1.TS1 SHRLEVEL REFERENCE/CHANGE
```

Application Table			
Col_1	**Col_2**	**Col_3**	**Col_4**
Data_1	Data_2	Data_3	Data_4
Data_1	Data_2	Data_3	Data_4

- Pending alter is recorded
- Object is in AREOR

ALTER TABLE → Online REORG



Application Table		
Col_1	**Col_2**	**Col_4**
Data_1	Data_2	Data_4
Data_1	Data_2	Data_4

DROP COLUMN Alter Statement

- A pending alter
  - Status is recorded in the SYSIBM.SYSPENDINGDDL catalog table
  - COLUMN_KEYWORD is „DROP“
- Advisory REORG Pending state(AREOR) is set for the table space
- Materialize drop column at the next table space level REORG
  - Online Reorg with SHRLEVEL CHANGE|REFERENCE on the entire tablespace materializes this pending alteration
  - Invalidation of all packages which reference the table

DROP COLUMN Syntax

- **ALTER TABLE …DROP COLUMN …RESTRICT**
  - RESTRICT is a required keyword – RESTRICT semantics means
    - Drop a simple column without any dependent objects (e.g. views, indexes, triggers, RI, unique/check constraints, row permissions, column masks, etc.)
  - Only one column per each ALTER statement
- After the drop is materialized, the column will be completely removed from the catalog and data. It will be as if the column never existed.
- Packages and dynamic cached statements that are dependent on the table are invalidated

DROP COLUMN Restrictions

- DROP COLUMN only applies to UTS only
- Can not drop the column
  - On MQT or table referenced by MQT (Materialized Query Table)
  - Tables with EDITPROC or VALIDPROC
  - Create Global Temp Tables
  - If the column is a Partitioned Key Column
  - If the column is a Hash Key Column
  - The last remaining column of the table
  - DOCID Column
  - ROWID Column with "GENERATED BY DEFAULT" or with a dependent LOB
  - Security Label Column

New SQL Codes

SQLCODE -195 (note – last remaining column, not the last column at the end)

LAST COLUMN OF *table-name* CANNOT BE DROPPED

Explanation

An attempt was made to drop a column using an ALTER TABLE statement. The column cannot be dropped from table *table-name* because at least one of the existing columns must be preserved when altering a table.

SQLCODE -196

COLUMN *table-name.column-name* CANNOT BE DROPPED. REASON = *reason-code.*

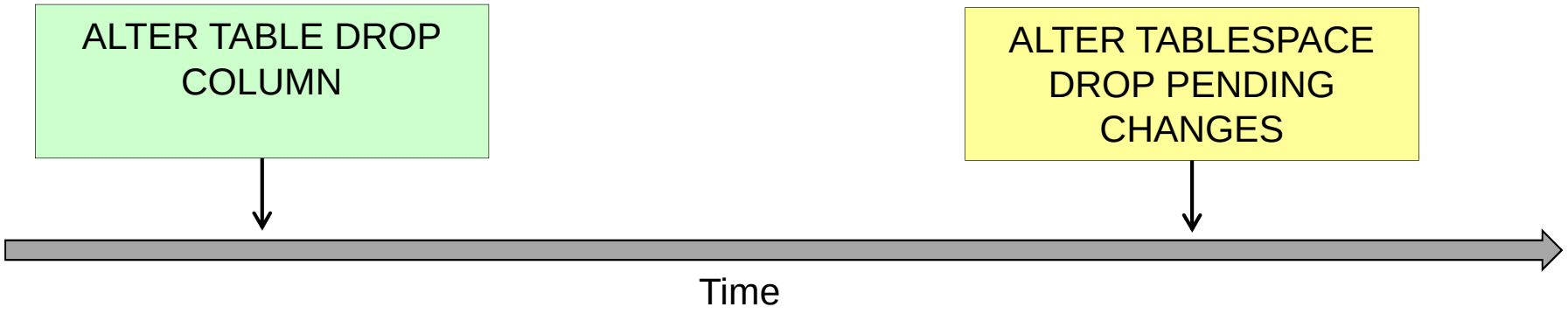
Explanation

The column cannot be dropped for one of the following reasons:

** See Information Centre or Manuals for the full list

Removing a Pending DROP COLUMN

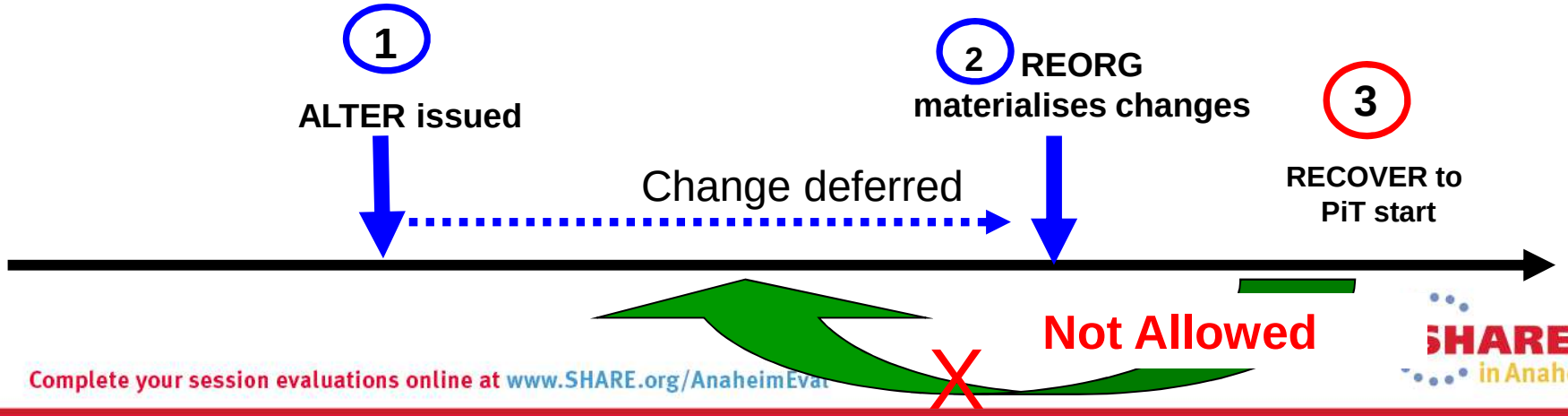
- If a pending ALTER needs to be cancelled before the materializing REORG...



- **NOTE: This removes <u>ALL</u> pending changes associated with the table space or any objects within the table space**

Impact on Utilities

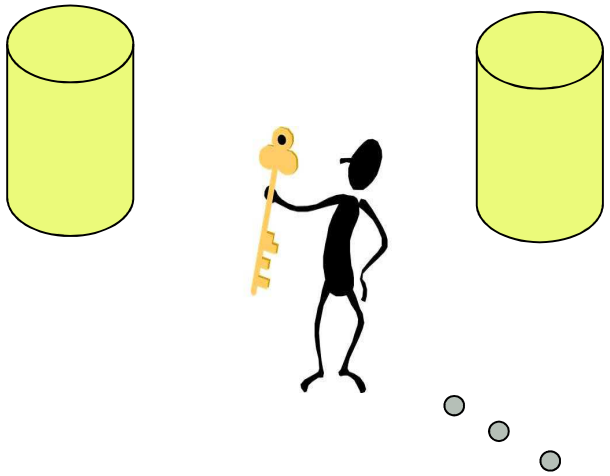
- RECOVER
  - Recover to PIT prior to materializing REORG is not allowed
  - SYSCOPY record with
    - *ICTYPE=A (=alter)*
    - *STYPE=C (=column)*
    - *TTYPE=D (=drop)*
  - DSNU556I and RC8
  - An inline copy will be taken by the REORG
    - *This single image copy is the only base for recovery*



Impact on Other Utilities - continued

- UNLOAD
  - UNLOAD will not be able to unload from an Image Copy which contains data for columns which no longer exist in the catalog because they were dropped.
    - DSNU1227I and RC8
- RUNSTATs
  - Should be run after materializing REORG
    - Ensures statistics are current for BIND/REBIND or PREPARE

Availability Enhancements Online Schema Changes



Online Alter Partition Limit Keys

Alter Table ... Alter Partition 2 Ending At 350 Inclusive

DB2 Catalog
SYSIBM.SYSTABLEPART

Part-ition	Logical_ Part	Limitkey_I nternal
2	1	200
3	2	~~300~~ 350
4	3	400
1	4	500

status	physical view	logial view	keys
	DSNUM 2	Partition 1	50, 125, 150, 175, 200
REORP	DSNUM 3	Partition 2	225, 250, 275, 300
REORP	DSNUM 4	Partition 3	325, 350, 375, 400
	DSNUM 1	Partition 4	425, 450, 475 500

Any Reorg including physical part 3 and 4 must run to remove the REORP status

- Prior to V11 behavior
  - Affected partitions are set to REORP.
  - These partitions could not be accessed.
  - Any REORG could be run to redistribute the data and remove the status.

DSNUM 2	Partition 1	50, 125, 150, 175, 200
DSNUM 3	Partition 2	225, 250, 275, 300, ***325, 350***
DSNUM 4	Partition 3	375, 400
DSNUM 1	Partition 4	425, 450, 475 500

DB2 V11 : Alter Table ... Alter Partition 2 Ending At 350 Inclusive

DB2 Catalog

SYSIBM.SYSTABLEPART

Part-ition	Logical_ Part	Limitkey_I nternal
3	2	300

SYSIBM.SYSPENDINGDDL

Option_ Value	Part-ition
...350	3

status	physical view	logial view	keys
	DSNUM 2	Partition 1	50, 125, 150, 175, 200
AREOR	DSNUM 3	Partition 2	225, 250, 275, 300
AREOR	DSNUM 4	Partition 3	325, 350, 375, 400
	DSNUM 1	Partition 4	425, 450, 475 500

Online Reorg including physical part 3 and 4 must run to materialize the Alter

SYSIBM.SYSTABLEPART

Part-ition	Logical_ Part	Limitkey_I nternal
3	2	350

- SYSIBM.SYSPENDINGDDL is empty

Materialization takes place in the SWITCH phase

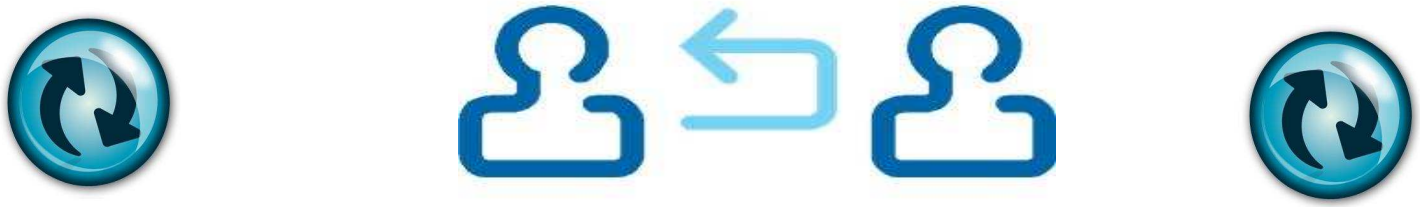
DSNUM 3	Partition 2	225, 250, 275, 300, ***325, 350***
DSNUM 4	Partition 3	375, 400

Summary – Alter Partition limit key

ALTER TABLE … ALTER PARTITION integer ENDING AT …

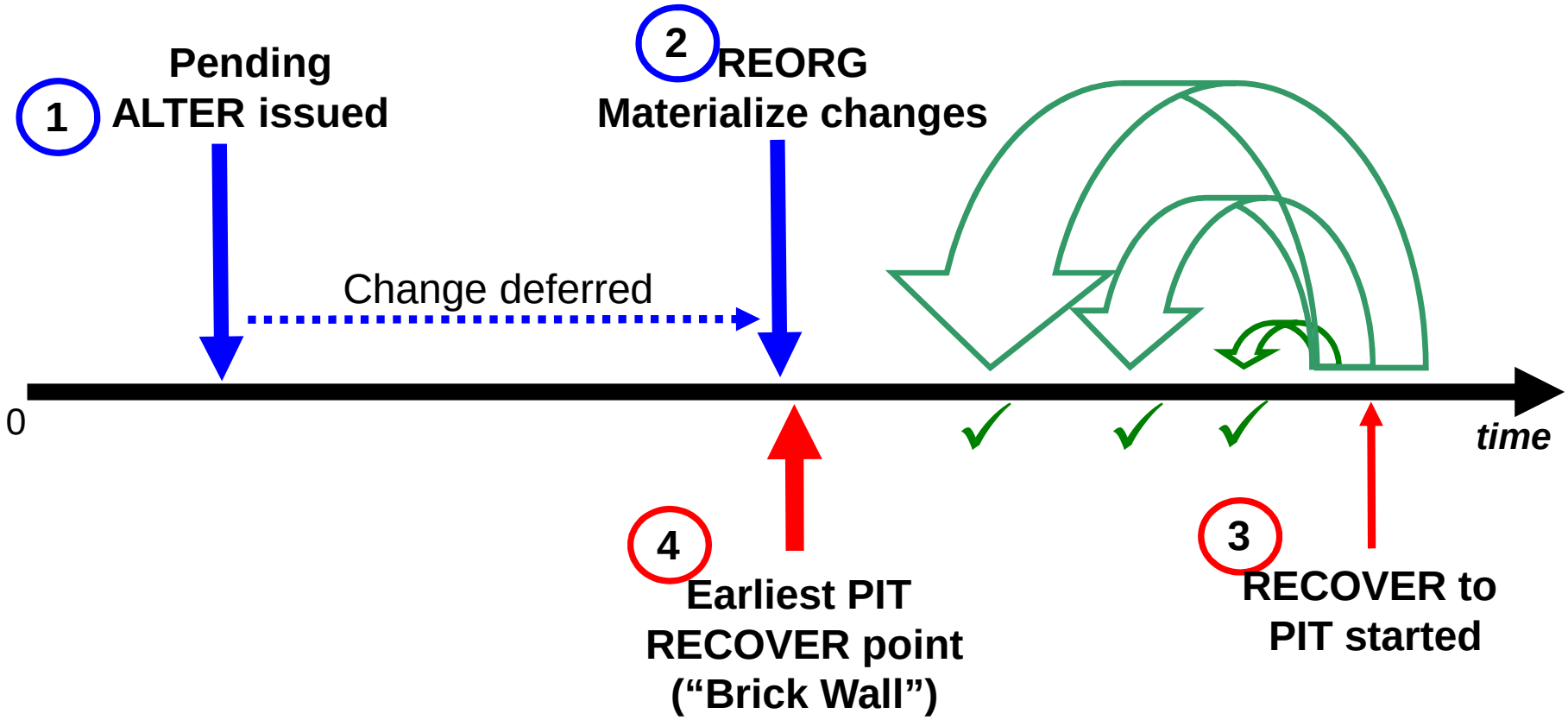
- Prior to V11 : Affected partitions are set to REORP
  - These partitions could not be accessed
  - Any REORG could be run to redistribute the data and remove the status
- V11 NFM, improves object availability
  - Alter limit key is treated as a pending alter
  - The affected partitions are set to AREOR
  - Online REORG **must** run to materialize the pending changes
    - REORG SHRLEVEL NONE does not materialize these changes
  - Supported table spaces types are:
    - UTS – partitioned by range (PBR)
    - Classic partitioned table spaces (table controlled partitioning)
      - *Objects, in which the partitioning is index controlled, must be first converted to table controlled partitioning*
  - The new limit keys are materialized in SYSTABLEPART in the SWITCH phase (new message DSNU2916I: PENDING ALTER LIMIT KEY value are being materialized).

Availibility Enhancements Online Schema Changes

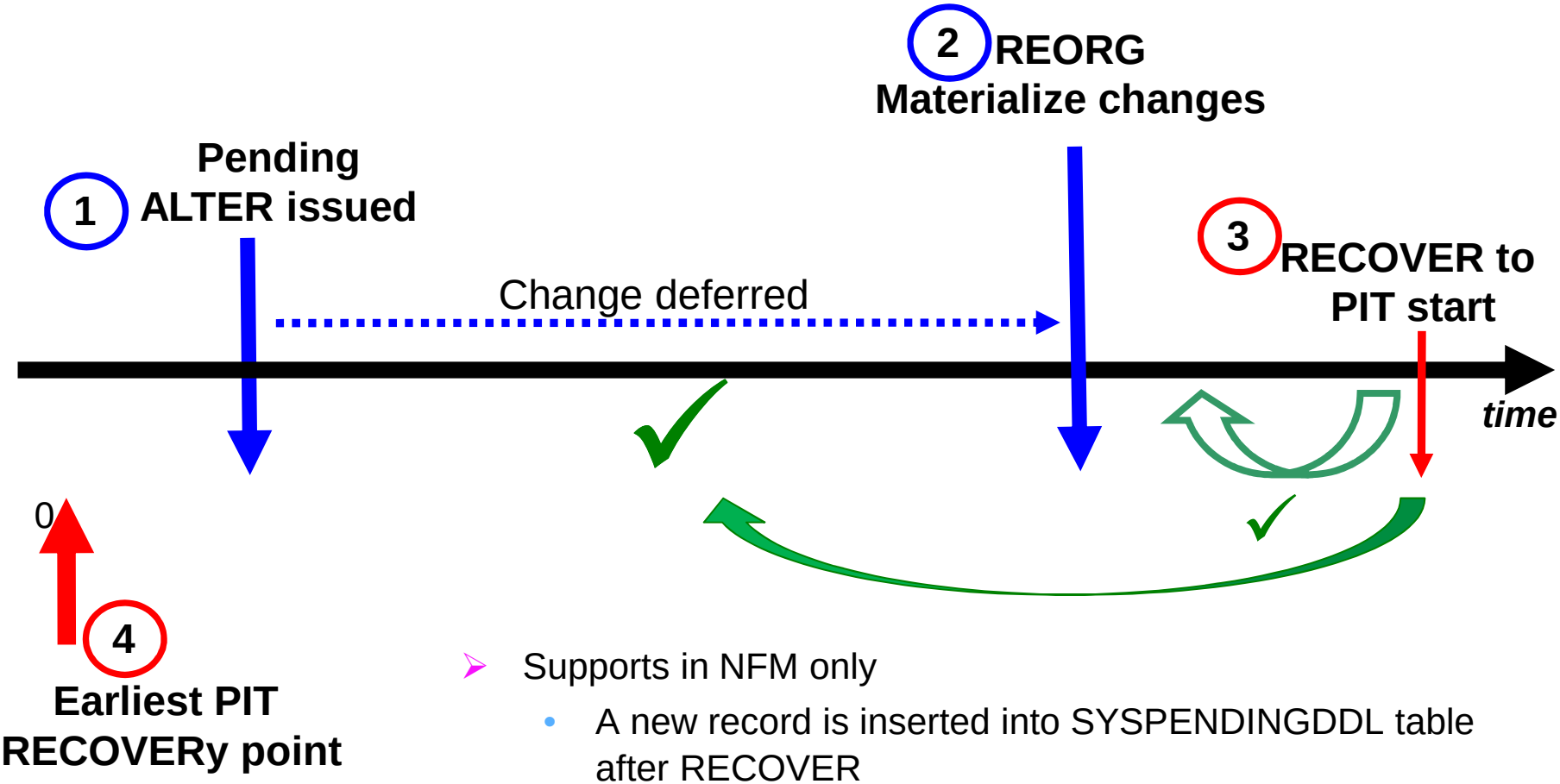


Allow PIT recovery before materialized deferred alters

DB2 10 PIT RECOVERy with Pending Alter



Enhanced PIT RECOVERy with Pending Alter in V11 NFM



- Supports in NFM only
  - A new record is inserted into SYSPENDINGDDL table after RECOVER
  - Table space is in restrictive status: REORG-pending

After PIT Recovery prior to REORG

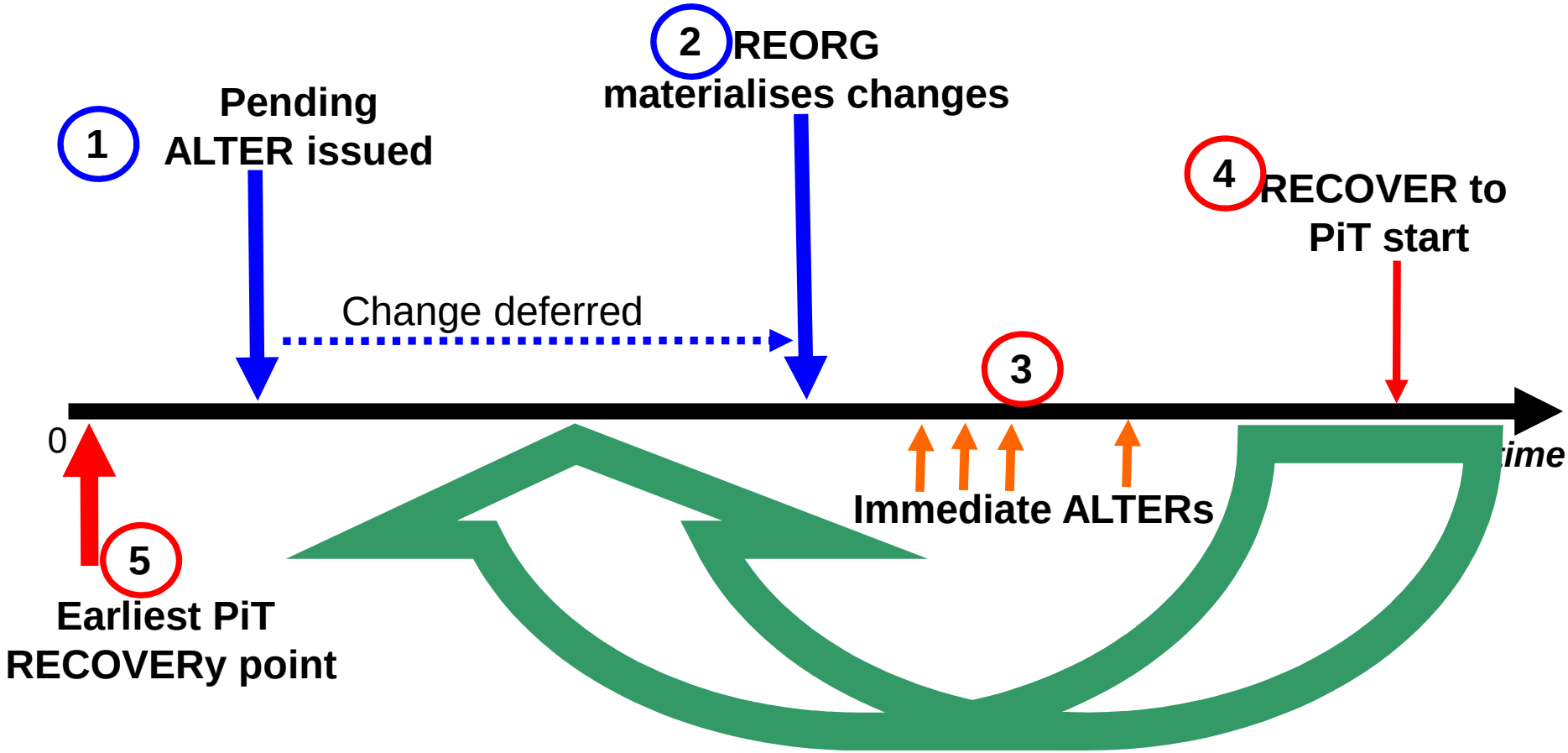
- At end of RECOVER to PIT prior to the materializing REORG:
  - REORG to finalize the PIT recovery process is MANDATORY
  - REORG must be on ENTIRE table space
  - SHELEVEL NONE is not supported
  - SHRLEVEL CHANGE is overridden by SHRLEVEL REFERENCE
- Before the subsequent REORG to materialize the RECOVER
  - No CREATE/ALTER/RENAME/DROP TABLE on the TS or AUX objects
  - All other utility job fails: DSNU933I (REORG required)
  - No other PITR during this period prior to subsequence REORG
- Only REORG/REPAIR DBD/REPORT RECOVERY or RECOVER to the same PIT are permitted
- Where there were pending changes on LOB table space:
  - First REORG the LOB table space
  - Then REORG the base table space

PITR of pending ALTER – Things to think about

- Pending alter can be dropped any time before REORG
- When recover pending alter to PITR after materializing REORG :
  - Recovering an index is not supported
  - Entire table space must be recovered
  - There must be NO other Pending changes when RECOVER starts

- Schema definition is not recoverable
  - **Only data is recovered to PIT**

PIT RECOVERy with immediate ALTERs



Pending ALTERs that has PIT Recovery support

- Table spaces type that has PITR capability after REORG materialized the pending ALTER :
  - LOB, XML and Partition by Range (PBR) Universal Table Space
- Pending Alter has PIT recover support :
  - Table space attribute alters (also with immediate ALTERs in the window between the materializing REORG and the PITR)

Pending ALTERs	PBR ts	LOB ts	XML ts
segsize	✓		✓
dssize	✓	✓	✓
bufferpool	✓	✓	
member cluster	✓		

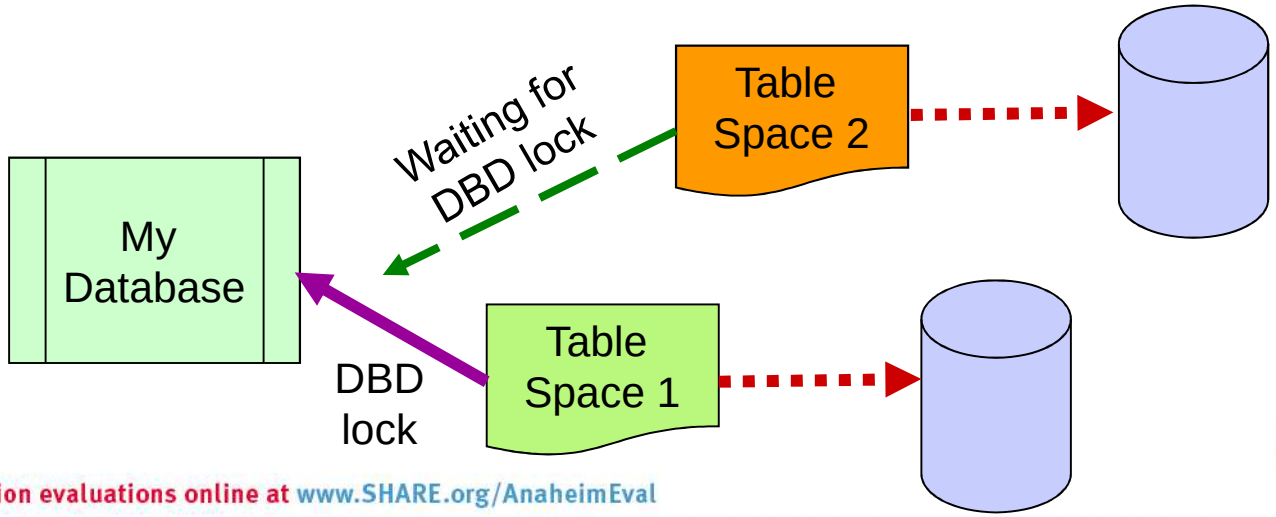
- RECOVER utility will issue message DSNU556I (recover cannot proceed), RC8 and terminate

Defer Define

CREATE TABLESPACE MyTS IN MyDB
NUMPARTS 454
.....
DEFINE NO
.....

Defer Define Enhancement

- Prior to V11
  - Many objects (Tables space) created with defer define option in the same DBD
  - Could encounter lock time out on DBD during the first insert/LOAD
    - DBD lock is not released until the commit
    - Long running UR holds the lock while other UR is waiting
- Solution in V11:
  - Release DBD lock as soon as data set is created and catalog is updated before UR commits
  - PM80967 retro-fit this function back to V10



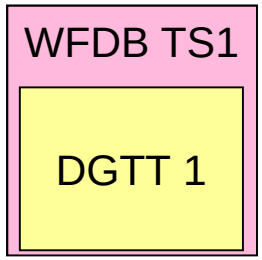
Workfile Enhancements

WORKFILE DATABASE (WFDB)

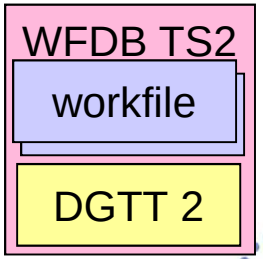
- Currently, each DB2 subsystem, or member of a data sharing group, has a single Workfile Database (WFDB).
  - E.g. DSNDB07, DB2PWORK, DB3PWORK, etc.
- WFDB used by DB2 for two purposes:
  - Declared Global Temporary Tables (DGTTs)
    - Declared by external applications
    - DB2 internal Static Scrollable Cursors
    - DB2 internal Instead of Triggers
    - Etc.
  - Non-DGTT temporary data held in 'workfiles'
    - Created Global Temporary Tables (CGTTs)
    - DB2 work data in support of:
      - *Sort*
      - *Materialized temporary results*
      - *Materialized views*
      - *Etc.*

WFDB Table Spaces (TS)

- WFDB can hold up to 500 table spaces (TS) for all types of temporary data
- Each WFDB TS can hold multiple DGTTs and/or workfiles
  - Number depends on size of the DGTTs and/or workfiles
- Data for a single workfile can reside in multiple WFDB table spaces
- A single DGTT cannot span multiple table spaces
  - DGTT limited to assigned TS only
  - For large DGTTs, secondary quantity allows table space to grow
    - For example UTS PBG



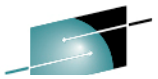
- If a DGTT is assigned to a TS also shared by workfiles, the DGTT can run out of space



WORKFILE DATABASE (WFDB) in V11

- DB2 11 improves application availability and performance by enhancing Workfile Database management
  - Instrumentation in DB2 Data Manager Statistics Trace record
  - DSNZPARM parameters
  - Warning messages at thresholds of agent or system use of Workfile Database (WFDB) storage
- Using these features, DB2 system administrators can monitor storage use and prevent application failures due to insufficient storage

Workfile Statistics trace records



- Add additional instrumentation data in DB2 statistics trace records (QIST) :
  - Total configured DASD storage for a work file database (QISTWSTG)
  - Current storage used and Maximum total storage ever used (include DGTTs): available in DB2 10
  - Maximum total storage even used by non-DGTT (QISTWFMXU)
  - Current storage used by non-DGTTs (QISTWFCTO)
    - New counter in V11
  - Total preferred configured storage for DGTTs (QISTDGTTSTG)
  - Maximum total storage ever used by DGTTs since DB2 is up (QISTDGTTMXU)
  - Current storage used by DGTTs (QISTDGTTCTO)
    - New counter in V11

DB2 11: DSNZPARMS, Instrumentation

- DB2 11 adds two new parameters to define space use thresholds
  - Issue a warning message when the total amount of in-use work file storage (including DGTT) has reached a storage shortage threshold (e.g.10% of total configured storage)
  - Once the warning message has been issued, DB2 will wait for 30 system check points before issuing another same warning message
  - WFSTGUSE_AGENT_THRESHOLD
    - Agent level space usage alert threshold
    - Online-updateable
    - Percentage in range 0 to 100
    - Default = 0
  - WFSTGUSE_SYSTEM_THRESHOLD
    - System level space usage alert threshold
    - Online-updateable
    - Percentage in range 0 to 100
    - Default = 90

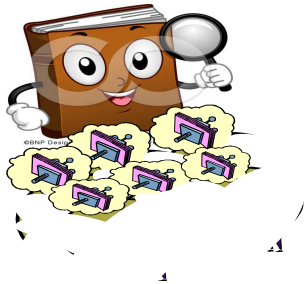
Agent

System

Compression Dictionary Availability for IFCID 306

User story:

- The replication products (e.g. Q-Rep) is unable to retrieve compressed rows when a new dictionary is built by LOAD/REORG.

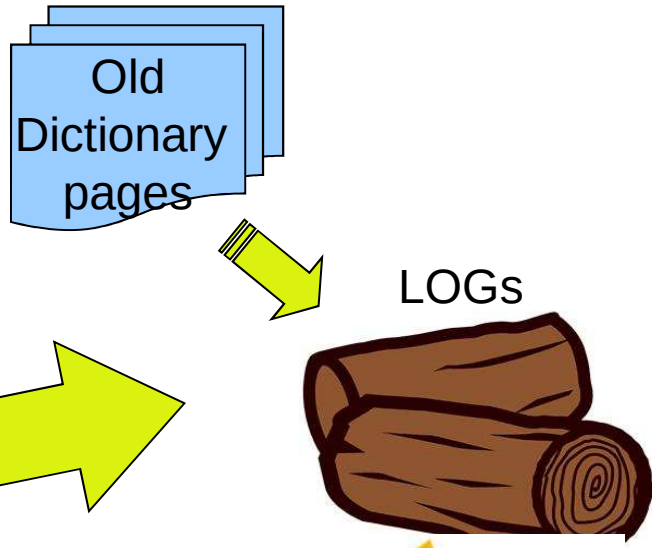


Behavior prior to Sequoia :

- The old dictionary is saved in memory for the data sharing member where LOAD or REORG operates.
- IFCID 306 will decompress logs with the dictionary in memory
- Dictionary in memory cannot be shared with different data sharing members

Solution for Availability on IFCID 306

- LOAD/REORG will write the old decompression dictionary to the log when a new dictionary is created by REORG or LOAD REPLACE – CDC tables only.



- IFCID 306 will decompress logs with the proper dictionary saved in logs.

- SYSCOPY records:
  - The RBA/LRSN value of the old dictionary will be recorded in SYSCOPY records.
  - ICTYPE = “J” – a new dictionary was built by REORG/LOAD
  - START_RBA – point to the log that contains an old decompression dictionary

Avoiding rebuild index after LPL/GRECP recovery

User story:

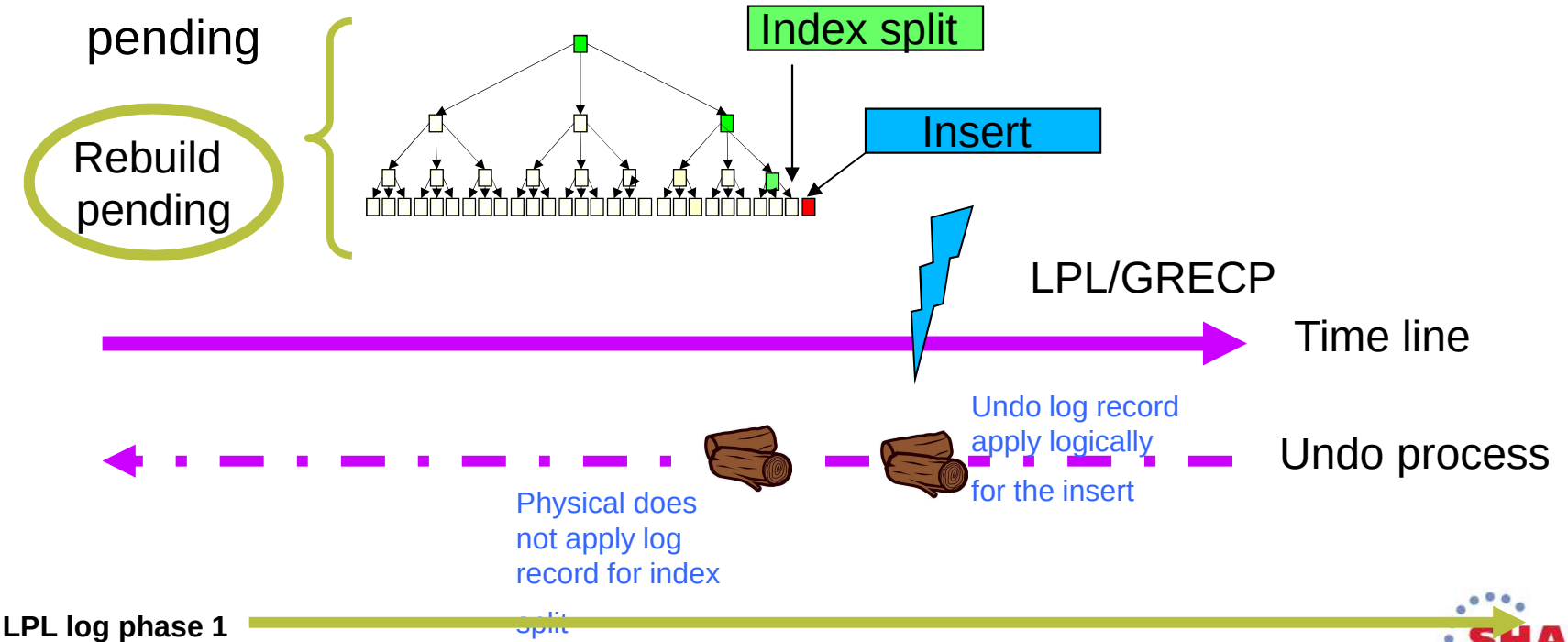
➢ If an index was marked in LPL/GRECP during an index split/merge, a subsequent LPL/GRECP recovery could mark the index in rebuild pending

- An index will be marked in rebuild pending when processing the first physical UNDO log (i.e. logs to rollback an incomplete index split/merge) after applying any logical compensation redo logs (i.e. LCLR)
- LCLR – logical compensation redo logs written when index pages were not accessible (e.g. LPL/GRECP)

Index rebuild pending after LPL/GRECP recovery

Prior to V11

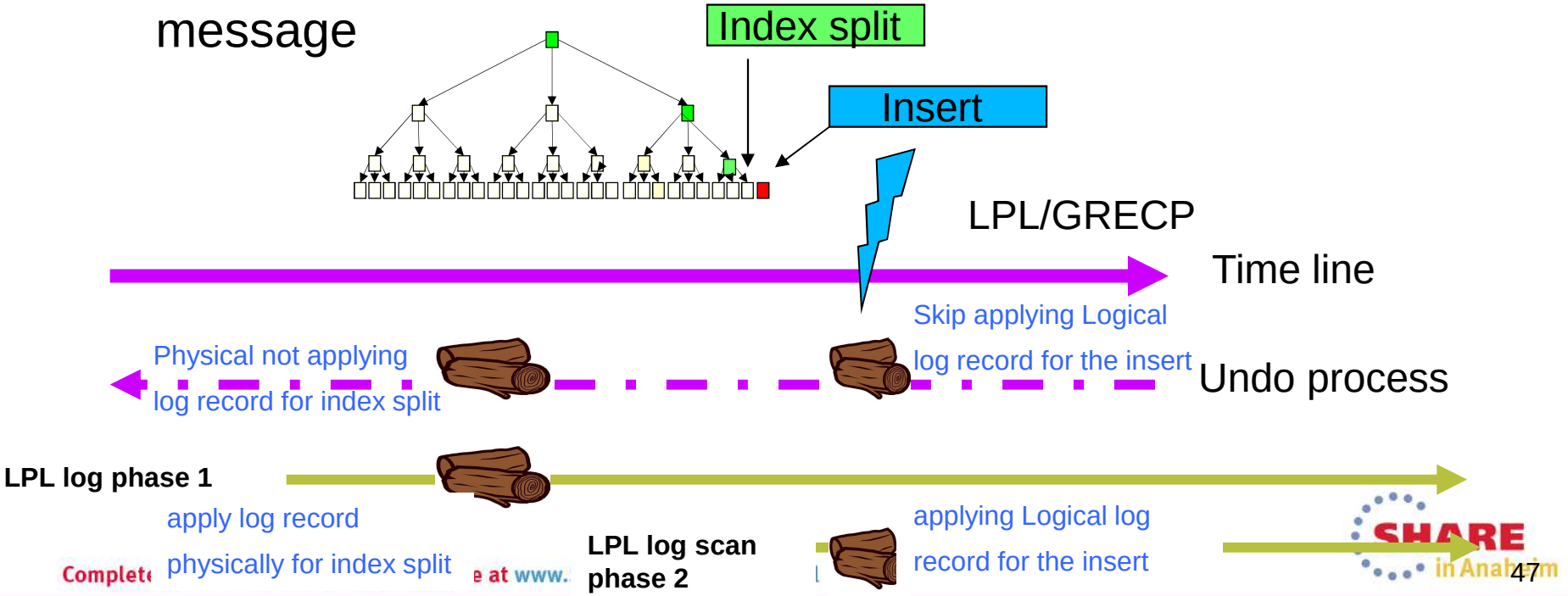
- If an index was marked in LPL/GRECP during an index split/merge, a subsequent LPL/GRECP recovery could mark the index in rebuild pending



Avoid rebuild index after LPL/GRECP recovery

Solution in V11:

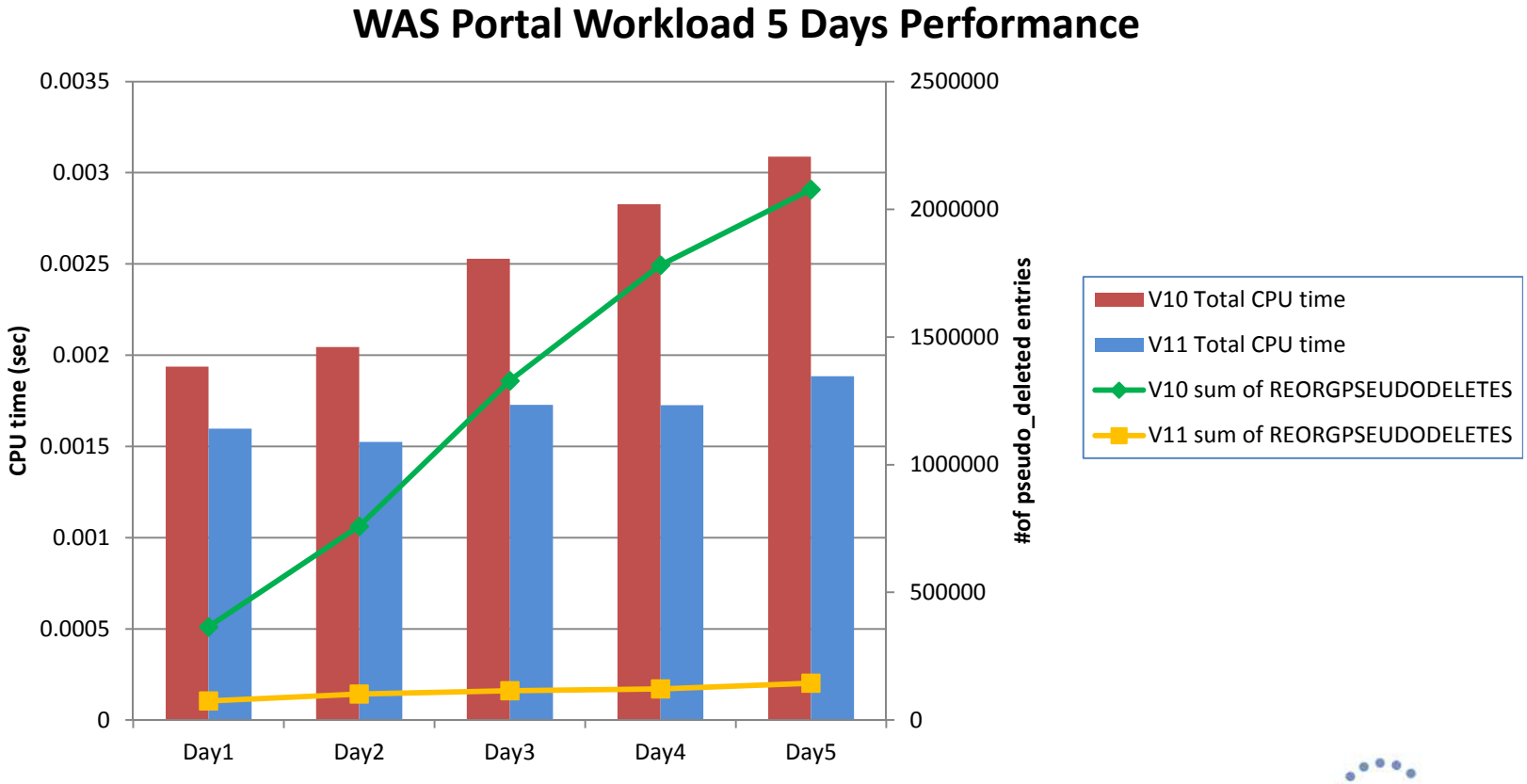
- LPL/GRECP recovery will defer process logical compensation redo logs after rollback an incomplete index split/merge
  - LCLRs will be processed by the second pass with a DSNI051I message



Pseudo delete cleanup

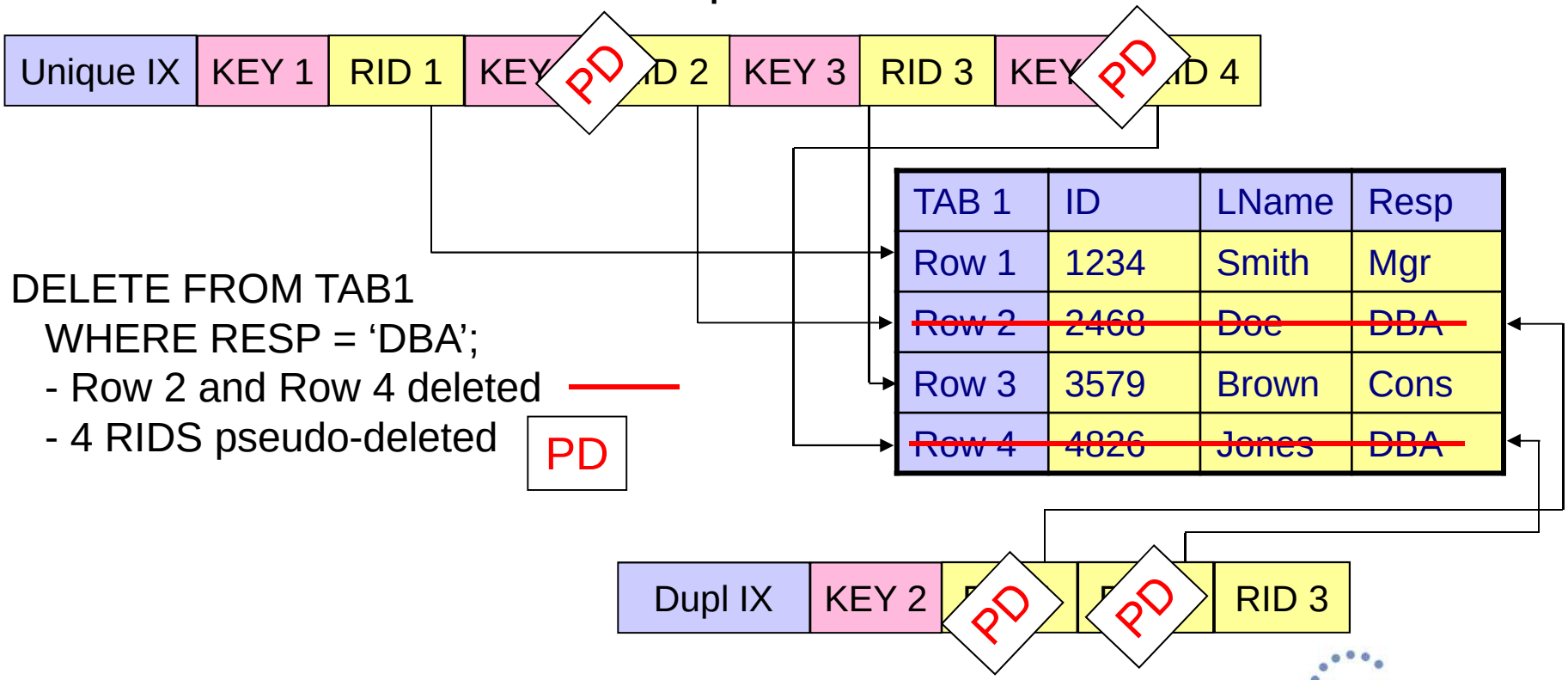
DB2 11 Auto Pseudo Delete Cleanup

- Up to 39% DB2 CPU reduction per transaction in DB2 11 compared to DB2 10
- Up to 93% reduction in Pseudo deleted entries in DB2 11
- Consistent performance and less need of REORG in DB2 11



Pseudo-deleted Index Entries

- Pseudo-delete process
  - When table rows are deleted, index RIDs are pseudo-deleted, unless the delete process has locked the entire table



Pseudo-deleted Index Entries

- Effects of pseudo-deletions
  - Index size grows with increasing number of pseudo-deleted index entries
    - More getpages and lock requests required
    - Increased CPU cost and possibly longer elapsed times for access via index search
  - Applications may encounter deadlocks and timeouts during INSERT/UPDATE/DELETE
    - Collisions with committed pseudo-deleted index entries
    - RID reuse by INSERT following DELETE => deadlock

Mainline logic that cleans up pseudo deletes

- Prior to DB2 for z/OS V11 , DB2 removes pseudo deleted entries during mainline operations
  - Insert / delete operations remove pseudo deleted entries from index pages
  - SQL with isolation level RR removes pseudo deleted entries.
- Pages that only contain pseudo-deleted index entries are called pseudo empty pages
  - DB2 attempts to clean up pseudo empty index pages as part of DELETE processing
- This is a partial solution, because
    - Performed under application thread, so extra application cost if aggressive cleanup
    - Cannot clean up uncommitted pseudo deletes
      - *The transaction that created the pseudo deletes can not clean them up*
      - *Has to rely on another transaction to clean up .*
- REORG INDEX removes pseudo empty index pages and pseudo deleted entries that were not cleaned up by the mainline processing

DB2 11: Pseudo-deleted Index Entry Cleanup

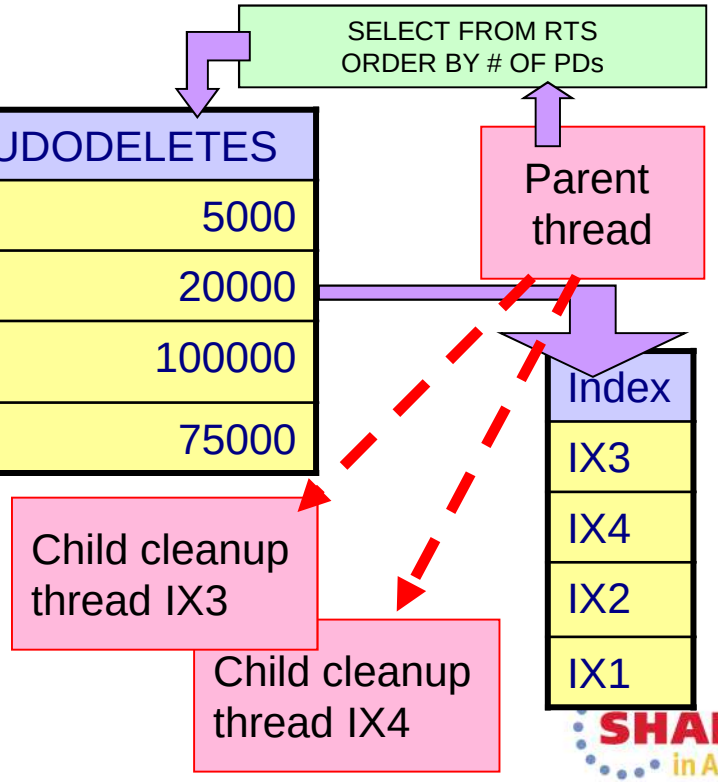
- DB2 11 solution, provided in CM
  - DB2 automatic cleanup of pseudo-deleted index entries in index page
  - DB2 automatic cleanup of pseudo empty index pages
  - DB2 11 continues to cleanup pseudo deletes and pseudo empty index pages as part of mainline processing
- Designed to have minimal or no disruption to concurrent DB2 work

DB2 11: Pseudo-deleted Index Cleanup

- Cleanup process
  - Cleanup is done under system tasks, which run as enclave SRBs and are zIIP eligible, the system doesn't have to be zIIP configured to use this new function.

SYSIBM.SYSINDEXSPACESTATS

NAME	…	NPAGES	…	REORGPSEUDODELETES
IX1	nn	100	xx	5000
IX2	nn	1000	xx	20000
IX3	nn	500	xx	100000
IX4	nn	2000	xx	75000



DB2 11: Pseudo-deleted Index Cleanup

- Parent thread (one per DB2 member) loops through RTS to find candidate page
  - Parent thread starts when DB2 starts and only comes down when DB2 is down or abend is encountered, CORRELATION ID 014.IDAEMK00
  - Checks RTS every several minutes.
  - Find the indexes with large number of pseudo deletes , check both the percentage of the pseudo deletes and the absolute number of pseudo deletes , the threshold is not documented and could be changed
  - Sort the list of indexes based on the number of pseudo deletes, the indexes with most pseudo deletes get cleaned up first.
  - The real cleanup is done by the child threads
- Child cleanup threads only clean up an index if it already is opened for INSERT, UPDATE or DELETE on the DB2 member, this is to avoid creating GBP dependency on indexes
- Child cleanup threads commits after cleaning up 64 index pages.

DB2 11: Pseudo-deleted Index Cleanup

- Mainline threads such as SQL using index scan and insert , delete , update that modify the indexes will add the index page with pseudo deletes into an in memory list.
- This list is maintained per index and used by the cleanup child threads to get the candidate index pages to be cleaned up.
- This avoids scanning the entire index to look for index pages with pseudo deletes and reduces the impact to system performance.
- It is possible that multiple members are cleaning up the same index at the same time. Each member has its own list of the pages to be cleaned up.
- If a page on the list has been cleaned up by another member, the cleanup thread will remove this page from the list and move to the next page
- Index page plock is used to maintain page consistency across different DB2 members.

DB2 11: Pseudo-deleted Index Cleanup

- Cost of cleaning up pseudo-deleted index entries
  - Delete processing of pseudo-deleted entries may reduce concurrency for index based access
  - Log volumes will increase as each pseudo-deleted index page or index entry cleaned up is logged
  - Use commit LRSN checking to determine if an index page is committed, no extra page/row lock acquired to check for commitness.
  - Need to hold index tree lotch when removing a pseudo empty page from the index tree
  - Release index tree lotch if a mainlain application is waiting to acquire the same lotch.

DB2 11: Pseudo-deleted Index Cleanup

- Potential disruption can be minimized by managing number of cleanup threads or specifying time when indexes are subject to cleanup
  - User can control the number of concurrent cleanup threads or disable the function using zparm INDEX_CLEANUP_THREADS (0-128)
    - *0 disables index cleanup*
    - *Value can vary between members of a data sharing group*
    - *Default is 10*
    - *Child clean up threads CORRELATION ID = 014.IDAEMK01, 02 ..*
  - Catalog table SYSIBM.SYSINDEXCLEANUP indicates when which indexes are enabled or disabled for cleanup

SYSIBM.SYSINDEXCLEANUP

- Control of cleanup to avoid disruption
  - Object level and timing of cleanup controlled through a new catalog table
    - Recommended to use for exceptions only
    - If table becomes too big, index cleanup performance may be impacted
  - Use SYSIBM.SYSINDEXCLEANUP to specify
    - Name of databases and indexes
    - Cleanup enabled or disabled
    - Day of week or day of month
    - Start time and end time
  - By default index cleanup is enabled for all indexes
    - If INDEX_CLEANUP_THREADS > 0 and SYSIBM.SYSINDEXCLEANUP table is empty
  - Sample to disable cleanup for all indexes

    INSERT INTO SYSIBM.SYSINDEXCLEANUP(DBNAME, INDEXSPACE, ENABLE_DISABLE, MONTH_WEEK, MONTH, DAY, START_TIME, END_TIME)
    values (NULL,NULL,'D', 'W', NULL, NULL, NULL , NULL);
  - Delay of up to 10 minutes before DB2 acts upon newly inserted row

SYSIBM.SYSINDEXCLEANUP

- SYSIBM.SYSINDEXCLEANUP and levels of control
  - Rows in SYSINDEXCLEANUP have effect on indexes as follows:

DBNAME	INDEX-SPACE	Level	Effect
NULL	NULL	System	All indexes in the system
Not NULL	NULL	Database	All indexes in this database
Not NULL	Not NULL	Index	This single index
NULL	Not NULL	invalid	N/A

  - If multiple conflicting rows apply to same index with overlapping time window
    - Rows on index level override rows on database level
    - Rows on database level override rows on system level
    - Same level: index cleanup disabled during overlapping window
- SYSIBM.SYSINDEXCLEANUP rows apply to all data sharing members

Using SYSIBM.SYSINDEXCLEANUP

- Examples
  - All index spaces in DB_1234 are enabled for cleanup on Sundays from 4:30 until noon, except
    - Index space IX_9876 is always disabled for cleanup. REORG INDEX requires specific window determined by DBA
  - All index spaces in DB_XYZ disabled for cleanup on Saturdays, and
    - Index space IX_ABC is disabled for cleanup on the 30th of each month from 1:30 to 7:30

SYSIBM.SYSINDEXCLEANUP

DBNAME	INDEX-SPACE	ENABLE_DISABLE	MONTH_WEEK	MONTH	DAY	START_TIME	END_TIME
DB_1234	NULL	E	W	NULL	7	043000	120000
DB_1234	IX_9876	D	W	NULL	NULL	NULL	NULL
DB_XYZ	NULL	D	W	NULL	6	NULL	NULL
DB_XYZ	IX_ABC	D	M	NULL	30	013000	073000

Pseudo-deleted Index Cleanup

- Instrumentation
  - IFCID 377 written once per index page being cleaned up
    - Includes DBID, PSID, partition number, page number
    - Indicates whether pseudo empty page or pseudo-deleted entries
    - Includes number of pseudo-deleted entries cleaned up
    - Not included in any trace class
      - *Must turn on IFCID 377 to monitor*
    - Report with OMPE RECORD TRACE
- DB2 11 Benefits
  - Reduce size of some indexes
    - Fewer getpages
  - Improve SQL performance
    - Lower CPU, less elapsed time
  - Reduce need to run REORG INDEX

Thank You

John Iczkovits

iczkovit@us.ibm.com